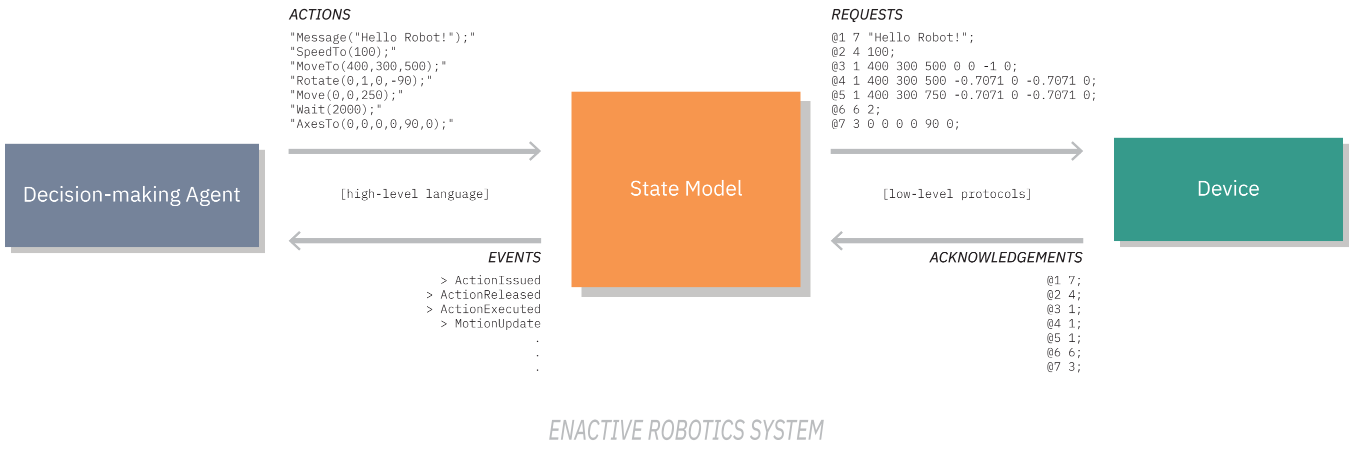
Robot Ex Machina

Jose Luis García del Castillo y López
Material Processes + Systems Group (MaP+S), Harvard GSD.
Generative Design Group, Autodesk.

A Framework for Real-Time Robot Programming and Control



```
"Message("Hello Robot!");"
"SpeedTo(100);"
"MoveTo(400,300,500);"
"Rotate(0,1,0,-90);"
"Move(0,0,250);"
"Wait(2000);"
"AxesTo(0,0,0,0,90,0);"
```

```
 > ActionIssued
> ActionReleased
> ActionExecuted
 > MotionUpdate
             .
             .
             .
```

```
@1 7 "Hello Robot!";
@2 4 100;
@3 1 400 300 500 0 0 -1 0;
@4 1 400 300 500 -0.7071 0 -0.7071 0;
@5 1 400 300 750 -0.7071 0 -0.7071 0;
@6 6 2;
@7 3 0 0 0 0 90 0;
```

```
@1 7;
@2 4;
@3 1;
@4 1;
@5 1;
@6 6;
@7 3;
```

1

1 The closed-loop interaction architecture of an *enactive robotics system*, the conceptual foundation of the real-time robot programming and control model of the *Robot Ex Machina* framework

ABSTRACT

Industrial robotic arms are increasingly present in digital fabrication workflows due to their robustness, degrees of freedom, and potentially large scale. However, the range of possibilities they provide is limited by their typical software control paradigms, specifically *offline programming*. This model requires all the robotic instructions to be pre-defined before execution, a possibility only affordable in highly predictable environments. But in the context of architecture, design and art, it can hardly accommodate more complex forms of control, such as responding to material feedback, adapting to changing conditions on a construction site, or on-the-fly decision-making.

We present *Robot Ex Machina*, an open-source computational framework of software tools for real-time robot programming and control. The contribution of this framework is a paradigm shift in robot programming models, systematically providing a platform to enable real-time interaction and control of mechanical actuators. Furthermore, it fosters programming styles that are reactive to, rather than prescriptive about, the state of the robot. We argue that this model is, compared to traditional offline programming, beneficial for creative individuals, as its concurrent nature and immediate feedback provide a deeper and richer set of possibilities, facilitates experimentation, flow of thought, and creative inquiry. In this paper, we introduce the framework, and discuss the unifying model around which all its tools are designed. Three case studies are presented, showcasing how the framework provides richer interaction models and novel outcomes in digital making. We conclude by discussing current limitations of the model and future work.

INTRODUCTION

During the last decade, we have witnessed an increased effort in exploring the new possibilities that industrial robotic arms could afford in the context of digital fabrication. However, several challenges still remain for larger adoption of these machines in digital fabrication. On the one hand, robots are notoriously hard to program: users are required to have significant domain expertise in order to be able to successfully use black-boxed environments to control these machines, with this knowledge often tied to a particular brand of industrial robot. This poses a great entry barrier that requires significant time investment to overcome, preventing novel users accessing these tools.

Most importantly, and even though industrial robots have become technically better over the years, it could be argued that the paradigms we use to program them have remained largely the same during the past half century (Peek 2016). Robot controllers, and their programming environments, perpetuate a classical machine control paradigm: *offline programming*. Under this model, all the motion planning, instructions, and logic must be pre-defined and translated to a program in the machine's native language, which will be executed without any further intervention from its creator. This is, for instance, the typical use case for any common 3D printing operation. Such paradigm is well suited for highly calibrated and predictable environments, where precise assumptions about their state can be done a priori. However, it is particularly constraining for systems with elevated degrees of uncertainty, or where robot behavior is mainly driven in response to evolving conditions around it. This is especially the case of robotic systems based on sensory input, environmental feedback, material properties, interactive installations, human-robot collaboration, etc. In this paper, we would like to argue that this model is particularly constraining from a creative point of view, as it prevents quick iteration and provides a rather shallow range of possible robot interaction models.

We hereby present *Robot Ex Machina*, a framework of computational tools for real-time robot programming and control. The framework is comprised of a collection of code libraries, plugins and stand-alone applications, all of them following the unifying principles of the enactive robotics model (García del Castillo 2019a). These tools are designed to integrate into programming environments popular among computational designers, makers and creative coders, and are publicly available in open-source repositories in order to maximize their educational and extensibility potential.

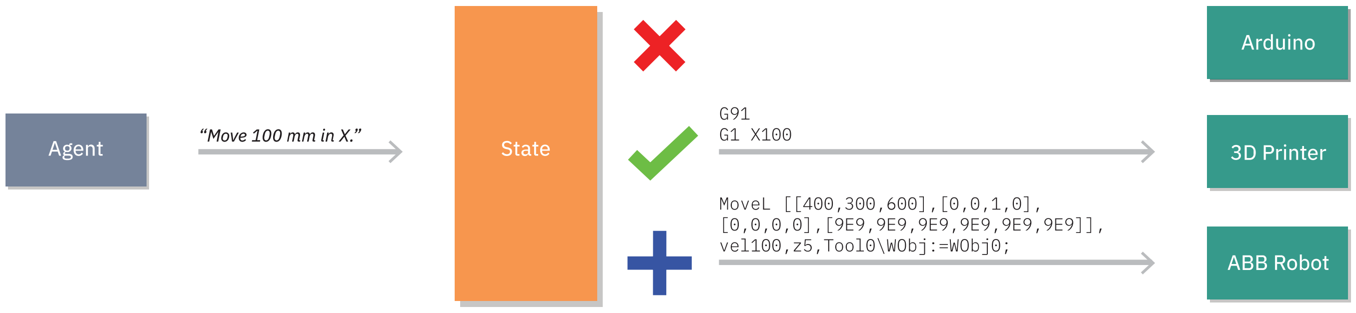
While it will be argued that the framework presented in this paper helps overcome some of the entry barriers present in typical robot programming environments, the main contribution of this research is to provide a systematic way to supersede the obsolete offline paradigm, allowing users to create robot control systems that are built with concurrent interaction at their core, and fostering programming styles that are reactive to, rather than prescriptive about, program execution. It will be argued that this framework enables a richer and deeper spectrum of possibilities for its users, in which its real-time nature and immediate feedback facilitates experimentation, flow of thought, and creative inquiry.

BACKGROUND

Several are the challenges still present in the democratization of robot programming and control. In industrial contexts, robot manufacturers usually provide highly proprietary frameworks specific to their products and the programming languages that run on them. These environments typically require significant training and domain expertise, and are oriented for highly specialized audiences in the engineering and manufacturing fields. Efforts have been made to open up the field of robotics to the research community (Quigley et al. 2009), but the target audience remains a reduced group of highly skilled individuals tied to engineering fields. Without the proper training and time investment, robotics is still out of reach to more general audiences.

Fortunately, some progress has been done in the last few years to bring creative communities closer to the world of industrial robots. The popularity of visual programming languages (VPL) for CAD/CAM workflows, such as Grasshopper3d, has led to the development of a large ecosystem of robot programming tools that integrate in this environment. Relevant examples of these tools are HAL Robotics (Schwartz 2013), KukaPRC (Braumann and Brell-Çokcan 2011), SuperMatterTools, Mussel (Johns 2014), TacoABB or Scorpion (Elashry and Ruairi 2014). While these tools usually streamline CAD to CAM processes, they are often brand-specific, requiring the rewriting of workflows with different tools to adapt them between systems. But most importantly, while some of these frameworks implement basic real-time communication, this functionality is reduced to automating program uploading to the robot, but does not involve live bi-directional feedback between machine and controller; they are fundamentally designed for *offline programming*.

Several authors have identified the limitations of this model, especially in the context of humans making things with



```
G91
G1 X100
```

```
MoveL [[400,300,600],[0,0,1,0],
[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]],
vel100,z5,Tool0\WObj:=WObj0;
```

2 An example of the action-state model of enactive robotics applied to different machines

machines. Proposals to palliate its effect include making the modeling and simulation phases concurrent (Landay 2009), advocate for models where robots complement conventional construction rather than substitute it (Bechthold 2010), propose interactive fabrication as the new making paradigm (Willis et al. 2011), or the exploration of the different levels of interaction units in personal fabrication (Mueller 2016). Ultimately, most authors agree that the big challenge to push our capacity to make things with machines lies in the software interfaces we use to mediate such interaction.

The will to break free of the constrains of the offline paradigm is clearly patent in recent attempts at enhanced robot fabrication systems via closed-loop control. Examples of such include projects accounting for tolerances in construction in near real time through "adaptive part variation" (Vasey, Maxwell and Pigram 2014), control of material feedback during production (Raspall, Amtsberg and Peters 2014) or sensory-informed autonomous robotic assembly (Jeffers 2016). Unfortunately, most of these projects relied on the development of bespoke solutions in the form of custom scripts and manually handled low-level machine communication. Sharif, Gentry and Sweet (2016) propose "adaptive control" to systematically incorporate humans on the loop on robotic fabrication processes, but their contribution stays at the conceptual level, with no technical counterpart. Shahmiri and Ficca (2016) describe a "model for real-time control of industrial robots," including technical architecture and sample implementations, but their contribution is specific to the Grasshopper3D environment and ABB robots.

The research presented in this paper aims to contribute a technical framework, backed by a conceptual model, in order to systematically enable concurrent programming and control of industrial robots.

THE ROBOT EX MACHINA FRAMEWORK

We hereby present *Robot Ex Machina*, a computational framework of software tools for real-time robot programming and control. The aim of the project is to provide a powerful, robust, and intuitive set of tools to access programming of industrial robotic arms, systematically situating *concurrency* at the cornerstone of its main control paradigm. The tools hereby presented address different groups of user levels, from novel users to advanced users, and can be integrated into common CAD/CAM environments, used to build applications or used as stand-alone ones.

Enactive Robotics

All the tools in the *Robot Ex Machina* framework are designed under the unifying principles of *enactive robotics* (García del Castillo 2019a; Figure 1), an action-state model for concurrent machine control. This paradigm is based on the theories of Bruner (1966) and Varela, Thompson and Rosch (2016), and situates action as the main cognitive unit of interaction between an agent and a system. The model is fundamentally based on the capacity of any decision-making agent—a deterministic algorithm, a neural network, a human—to interact in real-time with a system with the intention of controlling its behavior. In this context, an "action" is defined as a "request from the decision-making agent to change some of the properties of the controlled device." Actions must be:

- *Self-contained*: they embed all the information they need to convey their message
- *Platform-agnostic*: they contain no information about the device they are targeting
- *Universal*: can be applied to any device
- *Contextual*: their effect depends on the type of device that executes it and its state
- *Sequential*: they are executed in the order they were issued

Additionally, this paradigm relies on the central role of a *state* model, which contains a "concurrent representation of the changing properties of the devices participating in the system," and acts as the mediator between the agent and the machine. The state model must be:

- *Mutable*: should change dynamically to reflect changes in execution
- *Asynchronous*: be independent of any external runtime
- *Mediator*: perform translations between high-level actions and their machine representations
- *Transparent*: any agent should be able to query its properties at run-time

Figure 2 illustrates a conceptual example of this model. An agent requests the action "Move 100 mm along X axis" to the state model. The action contains all the information it needs to communicate the message but is agnostic to the system that will receive it. The state model receives the action and perform the necessary operations to translate it to a real-time instruction for a particular device. If this device is a 3D printer, the information contained in the action can be directly mirrored into to GCode. If the device is an ABB robot, the model needs to supply additional information based on the device's current state—orientation, configuration, velocity, zone, etc.—in order to generate a RAPID call. If the device is an Arduino board, the action can simply not be fulfilled.

This action-state model is particularly apt for the concurrent control of mechanical actuators, as it is capable of bridging the large-scale differences between the typically fast run time of a computational agent, and the slower execution time of mechanical actuators. Furthermore, due to its universality, it is applicable to any machine that performs motion in three dimensions, as well as apt to control different brands of robots under the same system. Finally, it provides a cognitive model that is easy and intuitive to learn, facilitating access to robot control to wider audiences.

Machina.NET

The core of the Robot Ex Machina framework is *Machina.NET*, a library for robot programming and control written in C#. The library is designed for advanced users familiar with the .NET framework, and can be used to create standalone applications, or tools that integrate into common CAD/CAM applications such as Grasshopper3D or DynamoBIM.

Machina.NET features a public application programming interface (API) that consists of simple English verbs denoting action requests, such as `Move`, `Rotate`, `Transform`, `Speed`, etc. In Figure 3, a simple "Hello Robot" program is represented which connects in real-time to an ABB robot, moves it to absolute coordinates `400`, `300`, `500` mm, traces a vertical `250` mm long line, and sends the robot back to a "home" position by setting the rotation of its six axes to `0`, `0`, `0`, `0`, `90`, `0` degrees.

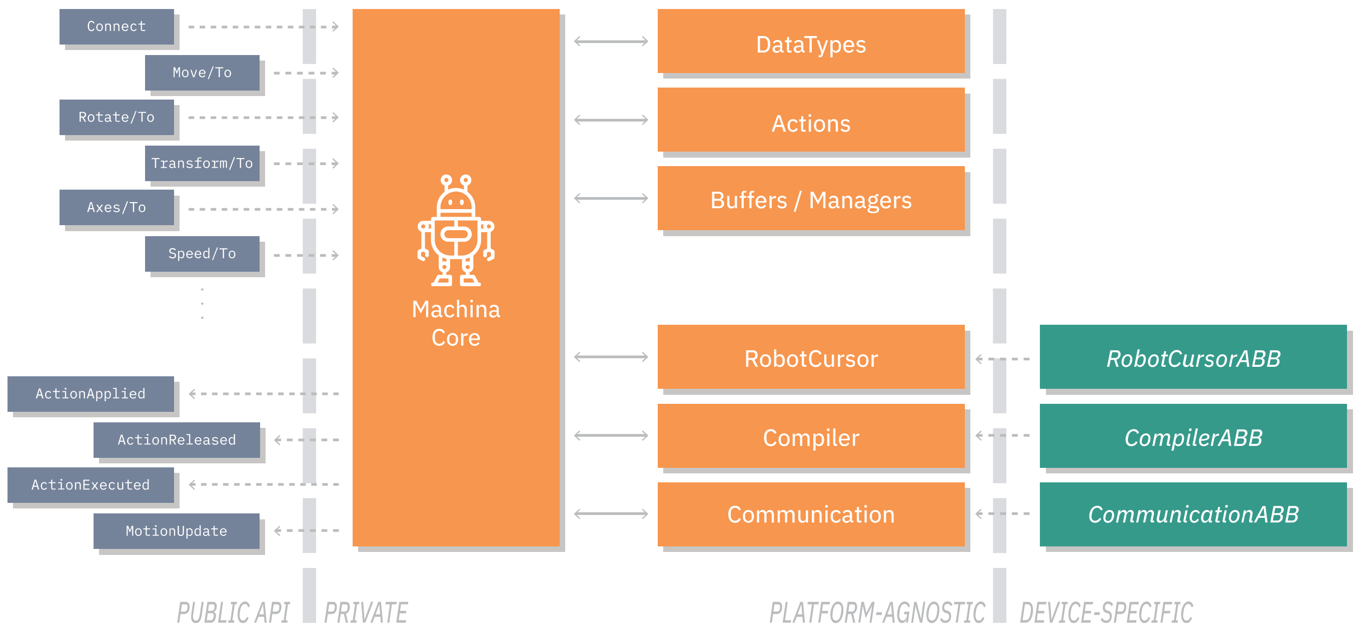
```
using System;
using Machina;
namespace Sample
{
    class MachinaSample
    {
        static void Main(string[] args)
        {
            Robot arm = Robot.Create("HelloRobot", "ABB");
            arm.ControlMode("online");
            arm.Connect("192.168.125.1", 7000);
            arm.Message("Hello Robot!");
            arm.SpeedTo(100);
            arm.MoveTo(400, 300, 500);
            arm.Rotate(0, 1, 0, -90);
            arm.Move(0, 0, 250);
            arm.Wait(2000);
            arm.AxesTo(0, 0, 0, 0, 90, 0);
            Console.WriteLine("Press any key to finish this program");
            Console.ReadKey();
            arm.Disconnect();
        }
    }
}
```

3 A sample "Hello Robot" program written in Machina.NET

The internal architecture of Machina.NET mirrors the action-state model described before, providing a public API of actions that can be used to control the device. An important characteristic of this API is that it provides absolute and relative flavors for most actions in the form of the `Action/To` syntax: `Move` is relative, while `MoveTo` is absolute. This is made possible by the central role of the state model, which keeps track of the evolution of the system, and is capable of translating an input relative action into an absolute machine instruction. The project's main repository contains an up-to-date full API of actions.

While Machina.NET can be used to generate a robotic program in traditional offline mode, the real potential of the library reveals itself when working in *online mode*. In this scenario, the library establishes a connection with a real or simulated controller—like RobotStudio—using a custom driver script. As soon as an action is requested, the user is notified in changes on the execution of this action via a set of asynchronous *events*, such as `ActionReleased`—when the action left the queue and was sent to the device—or `ActionExecuted`—when the device has finished executing that action. This API of notifications is extremely useful in designing applications or systems that react to changes in program execution.

The architecture of Machina.NET also makes it extremely modular and extensible. The core of the library is platform-agnostic, and incorporates a family of high-level descriptions for actions, as well as abstract classes as foundations to interface with physical devices. Machine states are tracked via a layered system of `Cursor` representations, which can be extended to accommodate different types of mechanical configurations. Similarly, the Compiler class can be extended to translate high-level actions to platform-specific code—GCode, RAPID, KRL, URScript, etc.—while low-level communication with



4 Overall architecture of the Machina.NET library

a particular controller can be implemented by inheriting from the `Communication` class (Figure 4).

A detailed description of the architecture and implementation of Machina.NET can be found on García del Castillo (2019b). This library is the core project of the Robot Ex Machina framework, incorporates all the technical architecture to make the action-state model possible, and serves as the technical foundation for all the other tools in the framework.

Machina Bridge

Machina.NET provides a powerful environment to create custom applications for real-time robotic systems. However, it requires a certain expertise in writing .NET applications, which may still constitute a great entry barrier to a large segment of the project's target users.

The *Machina Bridge* is a stand-alone, self-contained executable Windows application built with Machina.NET at its core, designed to provide immediate access to real-time robot control via a graphical user interface (GUI) (Figure 5), without the complexities of writing a desktop application. The interface features a series of input elements to customize the parameters for robot communication. Once successfully connected, users can use a command-line interface (CLI) to type Machina-styled actions either as single requests or lists of them. Upon sending the request for execution, the actions become buffered and released to the machine as soon as they get priority, with their execution tracked on the queue window. A status section, as well as a console window, give the user constant feedback about changes in execution and evolution of the system.

The Bridge proved to be a fairly successful tool at the earlier stages if this research. Despite the application having limited tools for program composition, its main value it two-fold. For novel users, the immediacy of typing an action and seeing it execute right away became a playful and powerful cognitive tool to understand basic concepts around robot programming. Moreover, as we will discuss in the next section, its most powerful feature is its capacity to act as mediator between the robot and any other client application willing to interface into it.

Other Programming Languages

The availability of Machina as a C# library only is a significant limitation, preventing users to access its functionality from other popular platforms. In order to extend robot control to other environments, users can still use the same conceptual action-state model using the Bridge as a gateway to the machine instead—hence its name.

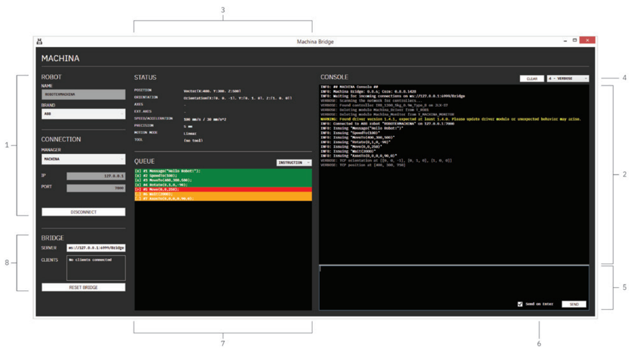
Aside from controlling a robot with the Bridge via the GUI, the application also provides access to its functionality to other applications via a background server. Any client willing to control the robot can use the WebSocket protocol, a standard in most modern programming languages, to connect to the Bridge server, and stream Machina-styled actions as string messages. Once received, the Bridge takes care of queuing those actions, parsing them into machine instructions, using low-level protocols to stream them to the controller, and tracking their progress (Figure 1). The client will in turn be notified of events rising over the execution of the actions, also as WebSocket string messages. This interaction model gives the Bridge the capacity to serve as a de-facto *stand-alone state model* in itself, able to serve as such to any other external application.

This cross-platform architecture allows the easy and fast creation of Machina-styled libraries for any programming language. The library only needs to feature a class as a thin wrapper around a WebSocket client, exposing a collection of methods matching the Machina API. The only function of the library is to translate the syntax and typing of the host platform to Machina-styled commands, and stream them to the Bridge via simple string messages. Optionally, the library may trigger responses after incoming messages from the Bridge via whichever idioms the platform uses: event handlers, method overrides, callbacks, etc. Such is, for instance, the common architecture behind *Machina for Processing*, *Machina for Python*, and *Machina for JavaScript* (Figure 6), all of them part of the Robot Ex Machina framework. This way, maintaining a library for the Machina ecosystem is reduced to ensuring compatibility with the main API, as all the rest of the real functionality can be delegated to the core .NET library, with the Bridge as its subsidiary.

CAD Environments

The flexibility of the model to provide a consistent interaction model in different environments makes it possible to provide high-level access to robotic manipulation for a variety of programming languages. However, many of these platforms lack the geometry processing tools necessary for complex CAD/CAM workflows, such as model slicing for 3D printing or toolpath generation for milling. The recent popularity of VPLs such as Grasshopper3D or DynamoBIM has democratized access to computational design workflows, and resulted in a myriad of robotic plugins that harness their geometry processing power.

Two implementations of the Machina architecture for VPLs are finally presented in this section: *Machina for Grasshopper3D* and *Machina for DynamoBIM*. Both projects are built on top of the Machina.NET library at the core, and distributed as plugins/packages that integrate into their respective hosts. They feature a collection of components/nodes that mirror the underlying Machina API of actions. In this way, programs can be composed by the translation of the platform's native geometry objects into Machina actions, and forming programs via their concatenation into lists. Such programs can be compiled into offline code for a target machine directly from the VPL,



```
POSITION             Vector[X:400, Y:300, Z:500]
ORIENTATION          Orientation[X:[0, 0, -1], Y:[0, 1, 0], Z:[1, 0, 0]]
AXES                 -
EXT.AXES             -
SPEED/ACCELERATION   100 mm/s / 30 mm/s^2
PRECISION            5 mm
MOTION MODE          Linear
TOOL                 (no tool)
```

```
[x] #1 Message("Hello Robot!");
[x] #2 SpeedTo(100);
[x] #3 MoveTo(400,300,500);
[x] #4 Rotate(0,1,0,-90);
[>] #5 Move(0,0,250);
[.] #6 Wait(2000);
[.] #7 AxesTo(0,0,0,0,90,0);
```

```
INFO: ## MACHINA Console ##
INFO: Machina Bridge: 0.0.6; Core: 0.8.0.1428
INFO: Waiting for incoming connections on ws://127.0.0.1:6999/Bridge
VERBOSE: Scanning the network for controllers...
VERBOSE: Found controller IRB_1200_5kg_0.9m_Type_B on JLX-S7
VERBOSE: Deleting module Machina_Driver from T_ROB1
VERBOSE: Deleting module Machina_Monitor from T_MACHINA_MONITOR
WARNING: Found driver version 1.4.1, expected at least 1.4.0. Please update driver module or unexpected behavior may arise.
INFO: Connected to ABB robot "ROBOTEXMACHINA" on 127.0.0.1:7000
INFO: Issuing "Message("Hello Robot!")"
INFO: Issuing "SpeedTo(100)"
INFO: Issuing "MoveTo(400,300,500)"
INFO: Issuing "Rotate(0,1,0,-90)"
INFO: Issuing "Move(0,0,250)"
INFO: Issuing "Wait(2000)"
INFO: Issuing "AxesTo(0,0,0,0,90,0)"
VERBOSE: TCP orientation at [[0, 0, -1], [0, 1, 0], [1, 0, 0]]
VERBOSE: TCP position at [400, 300, 750]
```

5

```
WebsocketClient wsc;
MachinaRobot bot;
void setup() {
  wsc = new WebsocketClient(this, "ws://127.0.0.1:6999/Bridge");
  bot = new MachinaRobot(wsc);
  bot.Message("Hello Robot!");
  bot.SpeedTo(100);
  bot.MoveTo(400, 300, 500);
  bot.Rotate(0, 1, 0, -90);
  bot.Move(0, 0, 250);
  bot.Wait(2000);
  bot.AxesTo(0, 0, 0, 0, 90, 0);
}
```

```
from machinaRobot import *
def main():
    bot = MachinaRobot("ws://127.0.0.1:6999/Bridge")
    bot.message("Hello Robot!")
    bot.speedTo(100)
    bot.moveTo(400,300,500)
    bot.rotate(0,1,0,-90)
    bot.move(0,0,250)
    bot.wait(2000)
    bot.axesTo(0,0,0,0,90,0)
main()
```

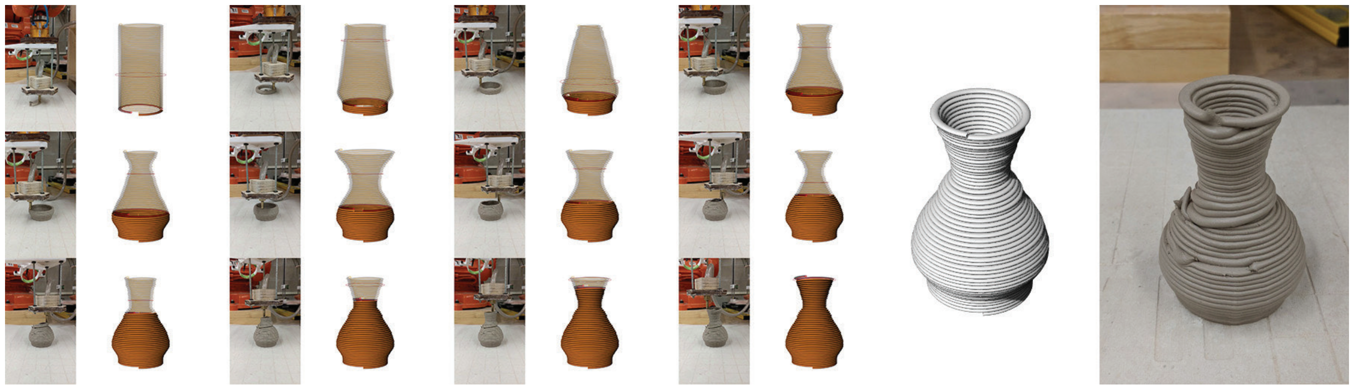
```
import WsClient from 'WsClient.js';
import Robot from 'machina.js';
let wsc = new WsClient("ws://127.0.0.1:6999/Bridge");
let bot = new Robot(wsc);
function start() {
  bot.Message("Hello Robot!");
  bot.SpeedTo(100);
  bot.MoveTo(400, 300, 500);
  bot.Rotate(0, 1, 0, -90);
  bot.Move(0, 0, 250);
  bot.Wait(2000);
  bot.AxesTo(0, 0, 0, 0, 90, 0);
}
start();
```

6

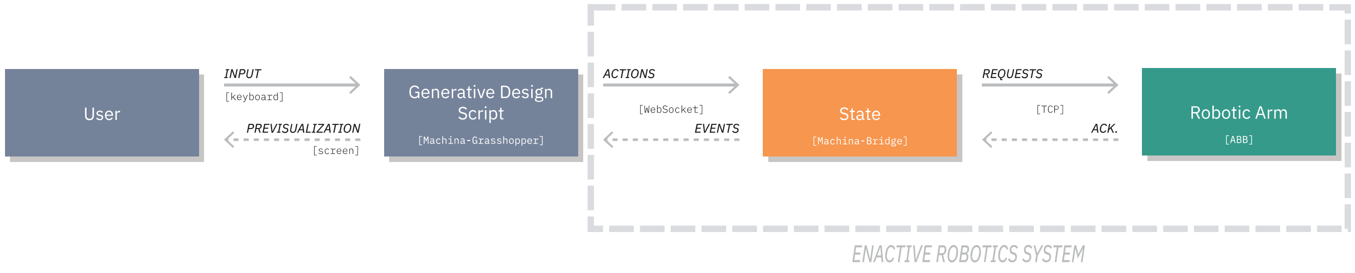
5 The Machina Bridge application, featuring: connection panel (1), status properties (3), console window (2) with log level (4), command-line input (5) with send options (6), queue window (7) and WebSocket server options (8)

6 The "Hello World" program in Figure 3, written in *Machina for Processing* (above), *Python* (middle), and *JavaScript* (below)

without external dependencies. However, and just like with the rest of the tools in the framework, real-time control can be enacted on a robot using the Bridge as the interface to it. The projects feature dedicated components to manage connection and streaming to the Bridge, via an internal WebSocket client, and can be set up to listen periodically to incoming notifications, triggering responses under certain circumstances. This way, a basic form of semi-responsive reactive programming can be developed in these environments which are otherwise not designed to accommodate execution loops.



7 Interactive modeling of a 3D printed clay object during print (left) and final result (right)



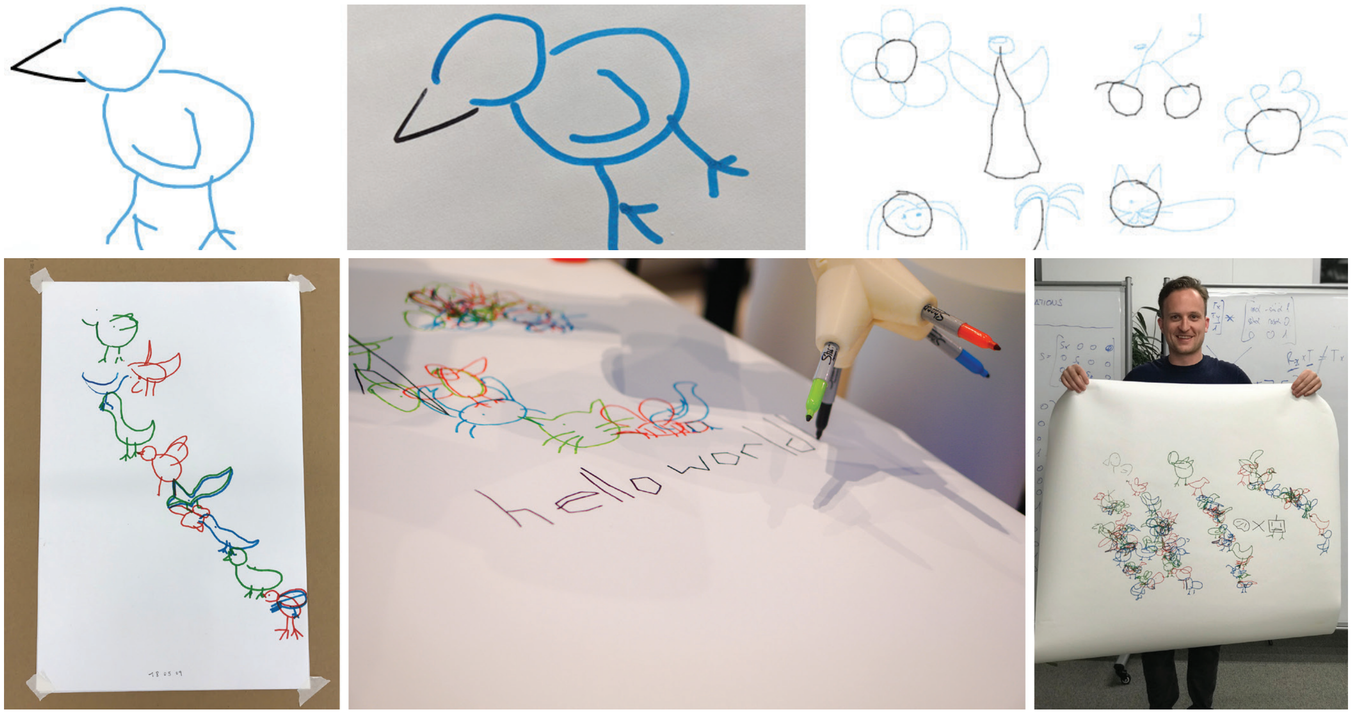
8 Closed-loop architecture of "Interactive 3D printing"

9 In "Tight Squeeze Pavilion," an on-site controller was able to perform changes over the baseline robotic program and plan drop-off motion (left), resulting in an intricate assembly built in three days by two robots and four human supervisors (right)

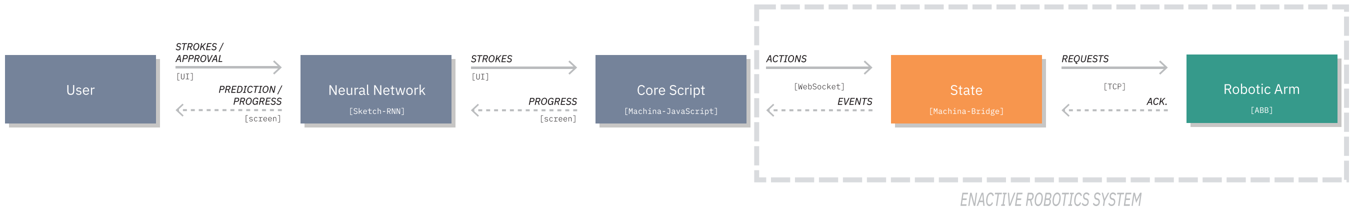
CASE STUDIES

The *Robot Ex Machina* framework has been successfully used for large scale robotic construction and manufacturing projects in conventional offline mode. These include the 2018 CEVISAMA Pavilion in Spain (Seibold et al. 2018) and the Greenbuild Pavilion in the USA (Hasan, Reddy and TsayJacobs 2019). However, the real power of the framework lies in its systematic capacity to enable concurrent control of a mechanical device, and the new scope of creative opportunities this possibility enables.

One of the most immediate disruptions this new paradigm can bring to conventional fabrication workflows is the introduction of real-time control in 3D printing. In Figure 7, the author created a simple Grasshopper3D script to 3D print a cylinder out of clay. As the script is continuously monitoring the print, it can stream layers sequentially as needed, while allowing room for concurrent human intervention. The creator can use the UI to navigate the model and modify the shape of the remaining layers, with the script imposing small restrictions based on the real-time state of the process to avoid failed prints. This way, the system allows for a closed-loop environment that involves a robot, a computational mediator and a human controller (Figure 8). Alothman et al. (2018) and Im, Alothman and García del Castillo (2018) further showcase this potential on "Responsive Spatial Print," a project where a spatial lattice of clay is 3D printed by measuring the deformation



10 Top: Sketches from the "RobotSketch-RNN" installation; a black stroke can be input to the system, and neural network will generate a suggestion in blue (left). Upon user approval, the robot will draw the sketch on paper (center). Different prediction models. Bottom: "Exquisite Corpse" piece.



11 System architecture of "RobotSketch-RNN" and "Exquisite Corpse": in the latter, stroke input was only possible upon the first sketch

of the clay as the material was deposited, and recalibrating the toolpaths of the next layer to counteract such distortion.

The idea of real-time human-robot collaboration was further explored during the "Tight Squeeze" workshop at the RobArch 2018 conference (Figure 9). Workshop leaders and participants built a large-scale pavilion out of 2x4 wooden studs by developing an interactive robotic construction system. The workflow started from a digital model of the pavilion, which generated simple robotic routines for member pick-up and drop-off in position. However, due to the complexity of the design, motion planning to avoid collisions would have been a huge challenge, difficult to solve with computational algorithms. Instead, the system relied on human supervisors able to input motion instructions via a set of video game controllers connected to the system, allowing real-time human guidance and tweaking over the baseline pick-and-place operation. Humans could pause motion execution, perform tweaks, and resume the procedure again, hence being able to manually avoid object collisions or account for material tolerances. This way, workshop participants were able to guide two industrial robots hanging from a gantry into building a pavilion with complex geometry and spatially intersecting planes.

The capacity of the Machina framework to systematically accommodate *any decision-making agent* as part of a robotic system opens up a wide spectrum of novel possibilities beyond manufacturing applications. During the "Mind Ex Machina" workshop at the SmartGeometry 2018 conference, participants explored creative opportunities at the intersection of robotics and machine learning. As a sample project, an interactive robot drawing application was designed by the workshop leaders, using machine learning as an interactive aid. Users could use a custom-built web application to select a sketch type—such as bird, cat or bicycle among others— and draw the first stroke of a doodle through a touch screen. A request would be sent to an HTTP server generating predictions from a Sketch-RNN model (Ha and Eck 2017), which had the capacity to generate a full drawing out that initial stroke (Figure 10, top). The prediction would be rendered on the screen, and upon approval, sent to the robot to be plotted on paper, using Machina for JavaScript and the Bridge (Figure 11). The collaborative relation between the human and the neural network showcases the potential of suggestive drawing as a new paradigm in human-computer creative interaction (Martínez 2017).

Workshop participant Sean Wallish ported the architecture of the system to Processing and its Machina counterpart, creating an art piece called "Exquisite Corpse". Drawing inspiration from the eponymous literary figure, he created an installation where, starting from an initial human stroke, the robot would draw Sketch-RNN doodles recursively using the last stroke of one sketch as the input for the next (Figure 10, bottom, and Figure 11), resulting in a collage of doodles that encapsulated the expression of the millions of humans who had previously trained the neural network.

LIMITATIONS AND FUTURE WORK

The framework is still under active development, and future work will go on addressing current limitations still present on the technical implementation. On the one hand, the core lacks a comprehensive library or robot models, along with dimensions, configuration, and joint limitations. This prevents the implementation of forward and inverse kinematic solvers, and delegates configuration solutions to the robot controller. Such limitation can be frustrating for the user, as the framework lacks the necessary information to prevent the robot from moving towards out of reach targets or through singularities; users can only find about such problems upon hitting robot execution errors. This lack also prevents certain translations between actions in relative and absolute coordinates, or between Cartesian and configuration spaces. Finally, and probably most importantly, it severely limits the capacity to generate visual simulations of programs before execution.

Additionally, working with robots—especially in a real-time fashion—can be dangerous for the user. Special attention will go into developing safety mechanisms that prevent users from requesting ill-defined actions, or continuous execution of programs with failed actions in them—a sort of default *strict mode* for novel users.

CONCLUSIONS

In this paper we presented Robot Ex Machina, a novel computational framework for real-time programming and control of industrial robotic arms. The framework provides a systematic way to command machines concurrently, and create robotic systems that can be inherently reactive, rather than prescriptive, about their behavior. The framework is inspired by the cognitive aspects of the enactive robotics model, and is designed to constitute an accessible entry point to robotics for novel users while, at the same time, providing a wider and deeper range of possibilities for advanced users. The different tools available in the framework were introduced, and their limitations were highlighted towards future development. Three case studies were presenting, showing the potential of this new paradigm to fundamentally change the role of robot control in experimental fields such as 3D printing, large-scale construction and art.

All the tools described in this paper are published as open-source projects, freely available to the community on *http://robotexmachina.com*. We hope that this contribution will help creative individuals supersede obsolete models of robot control and expand the standards of what can be done with machines and code.

ACKNOWLEDGEMENTS

The author would like to thank all the workshop participants, co-leaders and early adopters of the Machina framework for their patience and their invaluable contribution to the success of this project. Special gratitude goes to his advisors Martin Bechthold, Panagiotis Michalatos, and Stefanie Mueller, and to the whole MaP+S grop for their continued support and guidance. The research in this paper was partially made possible by the support received from the Real Colegio Complutense at Harvard. Finally, this project would have not been possible without the support of Autodesk Inc., the Generative Design Group and the Boston Technology Center Team.

REFERENCES

AlOthman, Sulaiman, Hyeonji Claire Im, Francisco Jung, and Martin Bechthold. 2018. “Spatial Print Trajectory.” In Robotic Fabrication in Architecture, Art and Design 2018, 167-80. Cham: Springer.

Bechthold, Martin. 2010. "The Return of the Future: A Second Go at Robotic Construction." Architectural Design 80(4): 116-121.

Braumann, Johannes, and Sigrid Brell-Çokcan. 2011. "Parametric Robot Control, Integrated CAD/CAM for Architectural Design." In Proceedings of the 31th Annual Conference of the Association for Computer Aided Design in Architecture, 242-251.

Bruner, Jerome S. 1962. On Knowing: Essays for the Left Hand. Harvard University Press.

Elashry, Khaled, and Ruairi Glynn. 2014. "An Approach to Automated Construction Using Adaptive Programing." In Robotic Fabrication in Architecture, Art and Design 2014, 51-66. Cham: Springer.

García del Castillo y López, Jose Luis. 2019a. "Enactive Robotics: An Action-State Model for Concurrent Machine Control." D.Des. Dissertation, Harvard University.

———. 2019b. "Machina.NET: A Library for Programming and Real-Time Control of Industrial Robots." Journal of Open Research Software 7(1): 27. Ubiquity Press. http://doi.org/10.5334/jors.247

Ha, David, and Douglas Eck. 2017. "A Neural Representation of Sketch Drawings." arXiv preprint arXiv:1704.03477v4.

Hasan, Hakim, Anish Reddy and Andrew TsayJacobs. 2019. "Robot Fabrication of Nail Laminated Timber." In Proceedings of the International Symposium on Automation and Robotics in Construction, Waterloo 36: 1210-1216.

Im, Hyeonji C., Sulaiman AlOthman and Jose Luis García del Castillo y López. 2018. "Responsive Spatial Print: Clay 3D Printing of Spatial Lattices Using Real-Time Model Recalibration." In ACADIA 2018, Proceedings of the 38th Annual Conference of the Association for Computer-Aided Design in Architecture, 286-93. Universidad Iberoamericana.

Jeffers, Michael. 2016. "Autonomous Robotic Assembly with Variable Material Properties." In Robotic Fabrication in Architecture, Art and Design 2016, 48-61. Springer.

Landay, James A. 2009. "Technical Perspective: Design Tools for the Rest of Us." Communications of the ACM 52(12): 80.

Martínez Alonso, Nono. 2017. "Suggestive Drawing Among Human and Artificial Intelligences." M.Des. thesis, Harvard University.

Mueller, Stefanie. 2016. "Interacting with Personal Fabrication Devices." PhD diss., University of Potsdam.

Peek, Nadya. 2016. Making Machines that Make: Object-Oriented Hardware Meets Object-Oriented Software. PhD diss., Massachusetts Institute of Technology.

Raspall, Felix, Felix Amtsberg, and Stefan Peters. 2014. "Material Feedback in Robotic Production." In Robotic Fabrication in Architecture, Art and Design 2014, 333-45. Springer.

Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. "ROS: An Open-source Robot Operating System." In ICRA Workshop on Open Source Software 3(3.2).

Seibold, Zach, Kevin Hinz, Jose Luis García del Castillo y López, Nono Martínez Alonso, Saurabh Mhatre, and Martin Bechthold. 2018. "Ceramic Morphologies. Precision and Control in Paste-Based Additive Manufacturing." In ACADIA 2018; Proceedings of the 38th Annual Conference of the Association for Computer-Aided Design in Architecture, 350-357. Universidad Iberoamericana.

Schwartz, Thibault. 2013. "HAL: Extension of a Visual Programming Language to Support Teaching and Research on Robotics Applied to Construction." In Robotic Fabrication in Architecture, Art, and Design, 92-101. Vienna: Springer.

Shahmiri, Fereshteh, and Jeremy Ficca. 2016. "A Model for Real-Time Control of Industrial Robots." In Proceedings of the International Symposium on Automation and Robotics in Construction, vol. 33. Vilnius Gediminas Technical University.

Sharif, Shani, T. Russell Gentry, and Larry M. Sweet. 2016. "Human-Robot Collaboration for Creative and Integrated Design and Fabrication Processes." In Proceedings of the International Symposium on Automation and Robotics in Construction, vol. 33. Vilnius Gediminas Technical University.

Varela, Francisco J., Evan Thompson and Eleanor Rosch. 1991. The Embodied Mind: Cognitive Science and Human Experience, rev. ed. 2016. Cambridge, MA: The MIT Press.

Vasey, Lauren, Iain Maxwell, and Dave Pigram. 2014. "Adaptive Part Variation." In Robotic Fabrication in Architecture, Art and Design 2014, 291-304. Springer.

Willis, Karl DD, Cheng Xu, Kuan-Ju Wu, Golan Levin, and Mark D. Gross. 2011. "Interactive Fabrication: New Interfaces for Digital Fabrication." In Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction, 69-72. ACM.

Jose Luis García del Castillo y López is an architect, computational designer, and educator. His current research focuses on the development of digital frameworks that help democratize access to digital technologies for designers and artists. Jose Luis is a registered architect and holds a Doctorate in Design and a Master in Design Studies on Technology from the Harvard University Graduate School of Design, where he is currently Lecturer in Architectural Technology along the Material Processes and Systems Group (MaP+S).